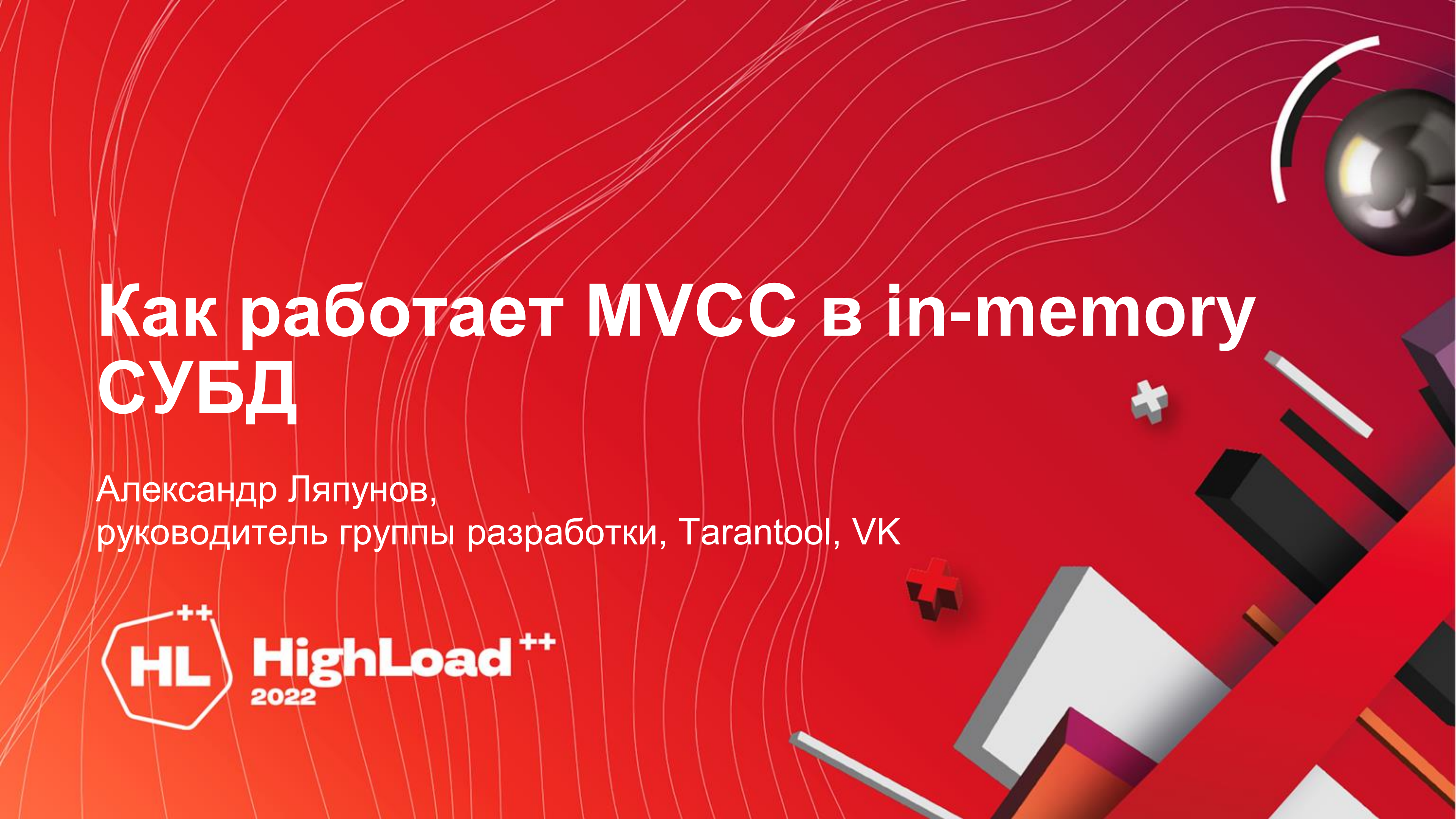


Как работает MVCC в in-memory СУБД
Александр Ляпунов,
руководитель группы разработки, Tarantool, VK
HL
HighLoad++
2022

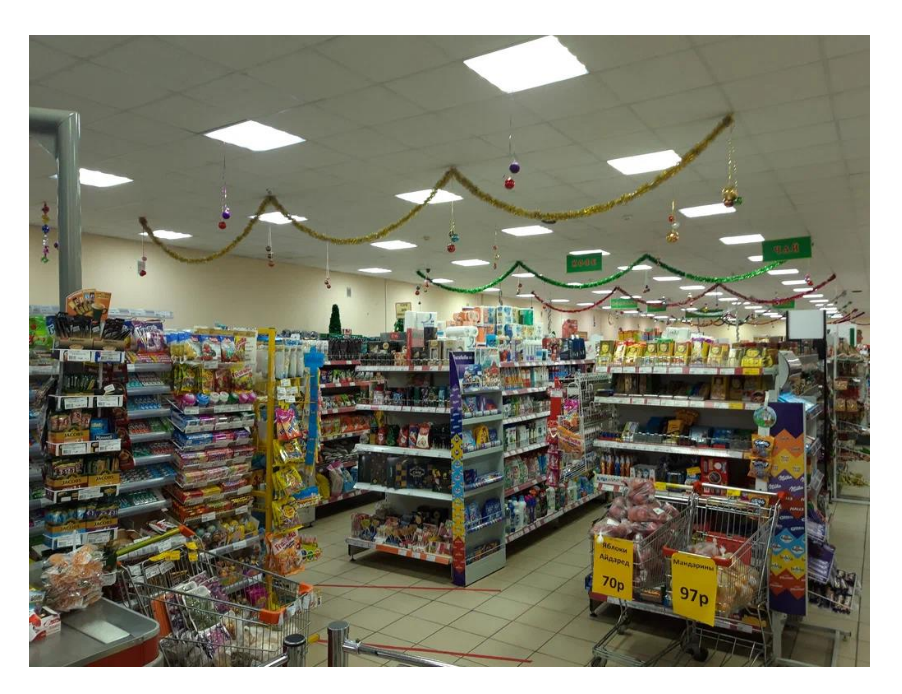
Содержание

01

Что такое и зачем нужен MVCC

Multi-version concurrency control

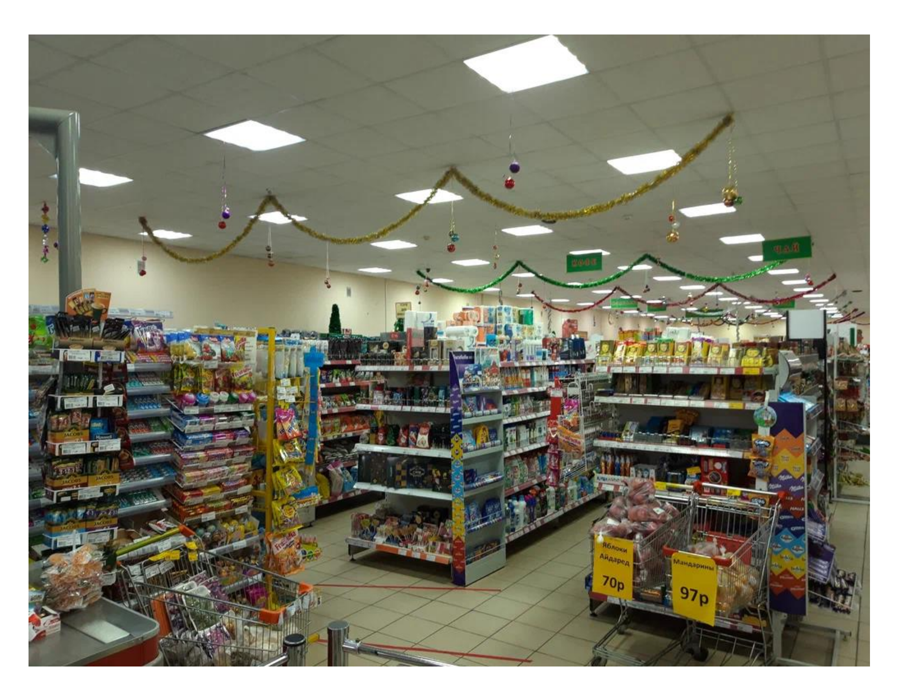
Шутка



Купи батон хлеба; если будут яйца – возьми десяток.



Шутка



Купи батон хлеба; если будут яйца – возьми десяток *яиц*.



Более сложная программа

Если будет белый хлеб - возьми, если нет - возьми половинку черного.

Если будет молоко, яйца и мука - возьми (будем блины печь), но если чего-то не будет - не бери ничего (не сможем блины печь).

Если будут чипсы со скидкой - возьми три пакета, но один пакет - в любом случае.





Человек - разумный исполнитель

Что будет, если что-то пойдет не так?

- Одновременно два исполнителя, а пакет молока – всего один.
- Скидка прекратится, пока исполнитель идет к кассе.
- …





Жизненный пример vs мир ИТ

Отличия мира ИТ

- Исполнитель – максимально тупой.
- Все происходит *очень* быстро.
- Объекты, как правило, эфемерны.
- Сложно на ходу “дописать программу”.





Жизненный пример vs мир ИТ

Что же общего

- Работа с изменчивым внешним миром.
- Мы хотим писать простые программы.





Борьба с изменчивостью

Как в реальном мире максимально упростить жизнь покупателю?

Может сделать так, чтобы никого не было? Варианты:

- Вход в магазин строго по одному (включая менеджеров).
- Подход к полке строго по одному.
- Подход к товару строго по одному.





Борьба с изменчивостью

Применимо ли это в мире ИТ? Да пожалуйста.

Мы только что изобрели *блокировки*!

- Блокировать всю базу данных.
- Блокировать таблицу.
- Блокировать одну запись.
- Блокировать диапазон записей.





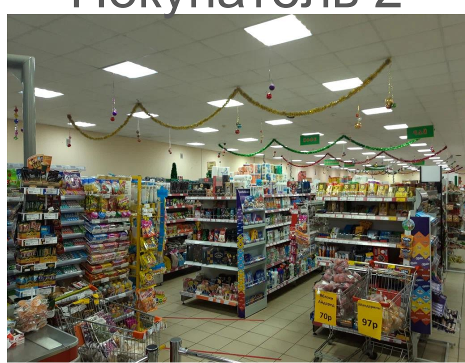
Multiversion concurrency control

А можно иначе? В ИТ – можно!

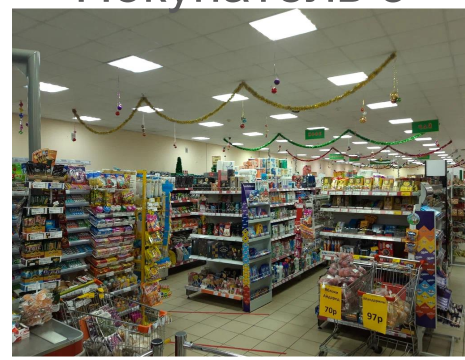
Покупатель 1



Покупатель 2



Покупатель 3



Быстро создаем каждому покупателю свой магазин.
На кассе *быстро* пересчитываем и дропаем неудачников.







Multiversion concurrency control

Особенности

- Каждый видит свою реальность.
- Каждый видит только свои изменения реальности.
- Нет ожидания.
- Кассир на выходе повторяет и сверяет все действия покупателя в *настоящей* реальности.





Multiversion concurrency control

Особенности

- Покупателей, у которых “не получилось” – нигде не фиксируем.
- С точки зрения кассовой книги все покупатели приходили строго последовательно.
- Покупатель “я только посмотреть” гарантированно видел какой-то момент в *настоящей* реальности.





MVCC в СУБД

Польза

- С правильным менеджером конфликтов (кассиром) – изоляция транзакций.
- Нет блокировок.
- Можно бесконечно долго рассматривать состояние на данный момент времени.



MVCC в СУБД

Недостатки

- Могут попросить "повторить", причем потенциально неизвестно сколько раз.
- На поддержку параллельных реальностей тратятся ресурсы.
- Подход сложнее в разработке.

Tarantool



02

Как (может быть) устроен MVCC в дисковых БД

Обзор

Дисковые БД

Особенности

- Начало берут с древних времен.
- Высокая стоимость чтения/записи случайных данных.
- Низкая стоимость последовательного чтения/записи.
- Дисковое пространство дешево.





Дисковые БД

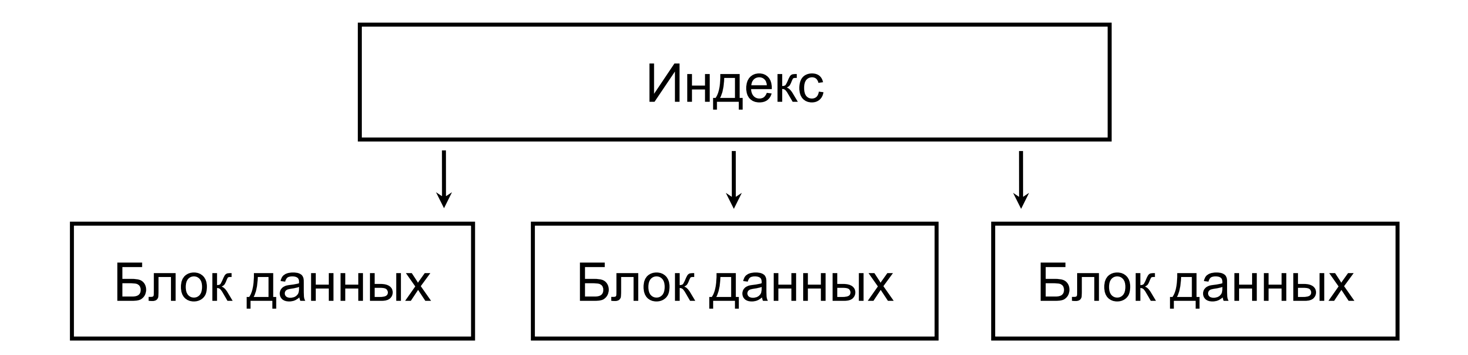
Особенности

- Необходима асинхронность работы транзакций.
- Ориентация на блоки данных.
- Тенденция к упорядоченному хранению.
- Общая нагрузка на БД не очень высока.
- Низкие требования к алгоритмам для локальных данных.





Дисковые БД



Дисковые БД

Таблички-указатели

Молочные

Хлеб

Чай/кофе

…
Батон нарезной – 100шт
…

…
Плюшка московская – 2шт
…







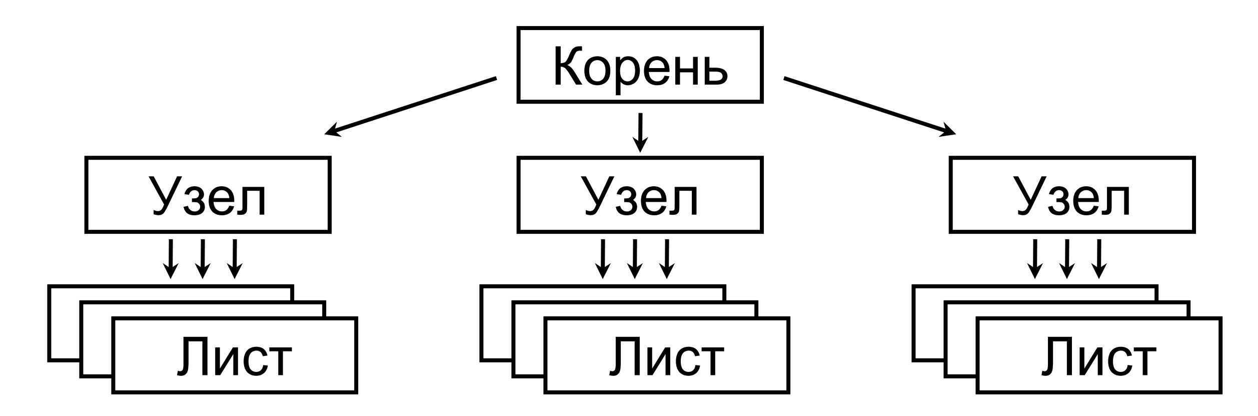
Дисковые БД

Индекс

Блок данных индекса	Блок данных индекса	Блок данных индекса



Сохранение копии данных: Б-дерево



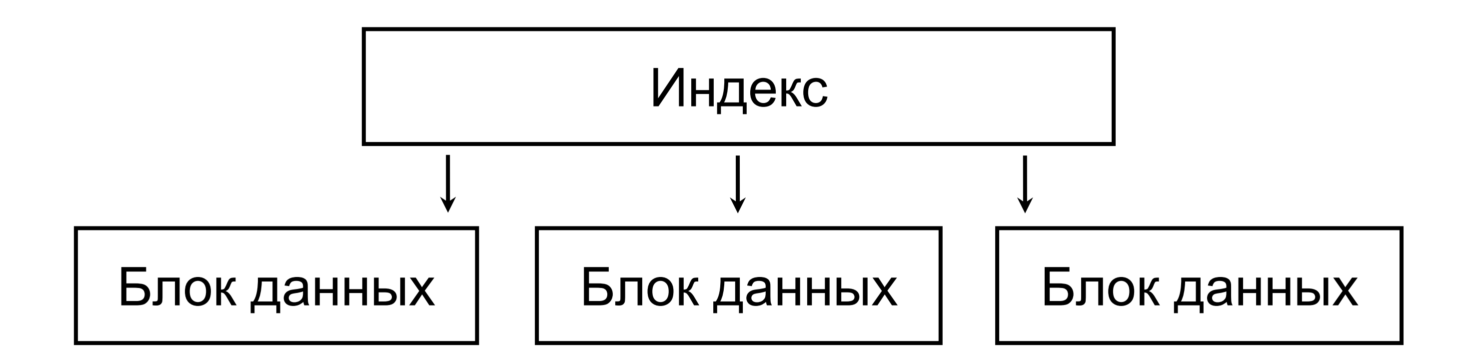


Как можно реализовать MVCC

Хранить копию данных.

Хранить историю изменений.

Сохранение копии данных



Сохранение копии данных

Главный индекс

Блок данных | Блок данных | Блок данных

Копия блока

Копия индекса

Удаление копии данных

Главный индекс

Блок данных

Блок данных

Блок данных

<мусор>

<мусор>

Сохранение копии индекса

Индекс

Блок данных индекса	Блок данных индекса	Блок данных индекса





Сохранение копии индекса

Индекс

Блок данных индекса

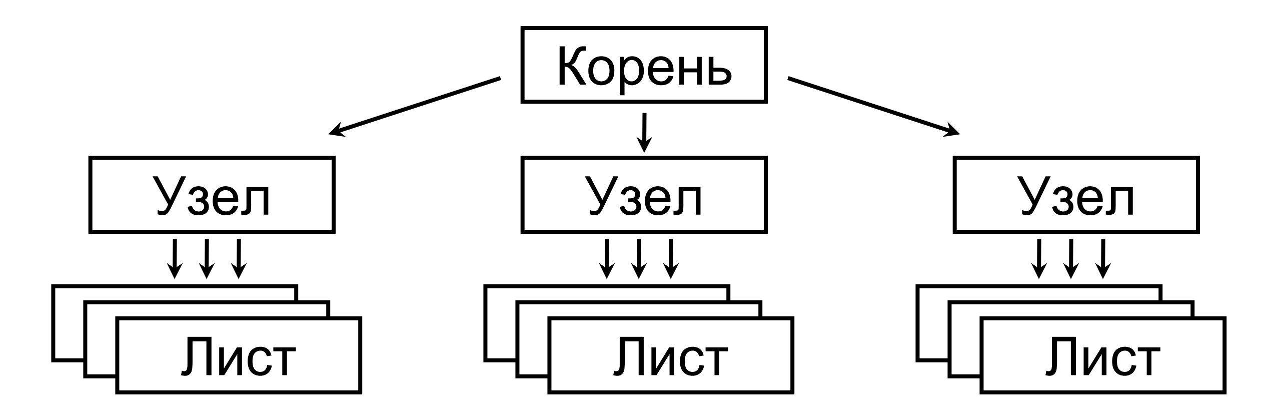
Блок данных индекса

Блок данных индекса

Копия блока данных

Копия индекса

Сохранение копии данных: Б-дерево





Сохранение копии данных: Б-дерево

Корень

Узел

Узел

Узел

Лист

Лист

Лист

Лист

Узел

Корень





Сохранение копии данных

Плюсы, минусы и свойства

- + Понятный и надежный механизм.
- + Минимум нагрузки на CPU.
- − На каждое изменение может быть скопирован блок данных.
- − Часто приходится копировать H блоков (высота дерева).





Сохранение копии данных

Плюсы, минусы и свойства

- − Возможна ошибка “недостаточно места” при удалении.
- − Долгая транзакция легко сожрет x2 данных.
- + Неизменную копию можно читать из тредов.
- + На диске место дешево.





Сохранение истории изменений

Основные принципы

- Вместо изменения данных – информация об изменении.
- Вместо удаления – добавление “надгробия”.
- При чтении разбирается история.
- История упорядочивается по времени и/или SN.





Сохранение истории изменений

Основные принципы

- Важно хранить всю историю одного ключа рядом.
- Фактически SN становится компонентом индекса.
- Периодически историю нужно “сжимать”.





Сохранение истории изменений

Таблички-указатели

Молочные | Хлеб | Чай/кофе

Батон нарезной:
6:00 было 100 шт,
10:45 стало 98 шт,
11:01 стало 97 шт.

Плюшка московская:
6:00 было 2 шт,
11:01 стало 1 шт.







Сохранение истории изменений: LSM-дерево

Индекс

Lvl 0: recent	Блок	Блок

Lvl 1: older	Блок	Блок	Блок	Блок	Блок	Блок

Lvl 2: oldest	Блок	Блок	Блок	Блок	Блок	Блок	Блок	Блок	Блок	Блок	Блок







Сохранение истории изменений

Минусы

- Довольно сложный механизм.
- Усложняется и удорожается обычное чтение.
- Хуже сочетается с многопоточным доступом.
- Возможна ошибка “недостаточно места” при удалении.



Сохранение истории изменений

Плюсы

- Экономичный для диска метод.
- Идеально подходит для LSM-деревьев.



03

Как (может быть) устроен MVCC в in-memory БД

Плюсы и минусы

БД в памяти

Особенности

- Относительно новое веяние.
- Много наработок своровано из дисковых БД.
- Память работает очень быстро.
- Работа с указателями.





БД в памяти

Особенности

- Память очень дорогая.
- Предполагается высокая нагрузка на БД.
- Высокие требования к алгоритмам.
- Надежность данных (запись на диск) никто не отменял.





БД в памяти

Пример адаптации Б-дерева для in-memory БД

- Нет данных (в т.ч. ключей), только указатели на данные.
- Специальный аллокатор.
- Трансляция адресов.
- Б*-дерево.





MVCC для БД в памяти

Способы реализации

- Адаптация алгоритмов из дисковых БД.
- Применение алгоритмов со случайным доступом к памяти.
- Запретить асинхронность.





Запретить асинхронность

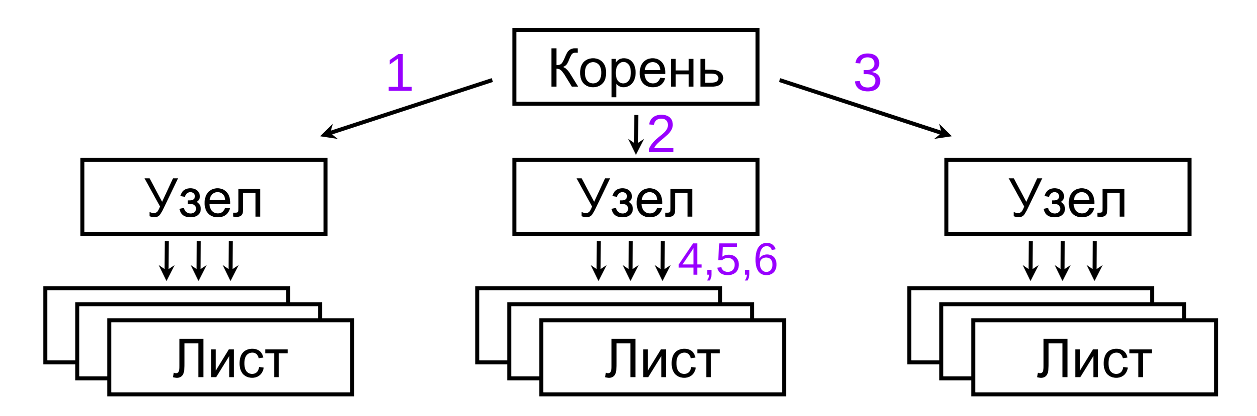
Вход - строго по одному!

- Каждая следующая транзакция не начинается, пока не закончится предыдущая.
- В транзакции запрещено всё сложное, например I/O.
- После commit и начала записи в WAL – транзакция видна.
- Сбой WAL может привести к грязным чтениям.
- Это – не MVCC.





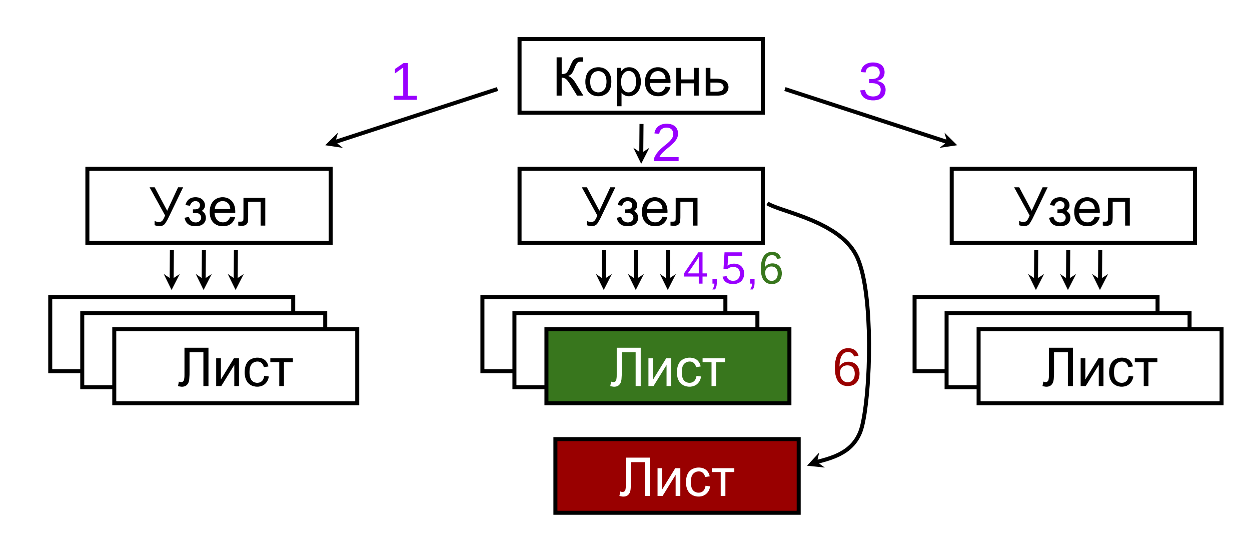
Сохранение копии данных: трансляция адресов







Сохранение копии данных: трансляция адресов







Сохранение копии данных: matras

Matras - Memory Address TRanslation Allocator (s?).



- Аллокатор с трансляцией адресов и CoW.
- MVCC для любой структуры данных.
- Аллоцирует блоки заданного размера (2^n).
- Для *данного* читателя отображает ID блока → блок.
- Подробнее - по ссылке в QR-коде.





Сохранение копии данных: read view

Опять специальный аллокатор

- Аллокатор общего назначения.
- Объект всегда доступен по указателю.
- Откладываем фактическое удаление объекта (если надо).
- Накладные расходы – uint32_t в каждом объекте.





Сохранение копии данных: read view

создание удаление

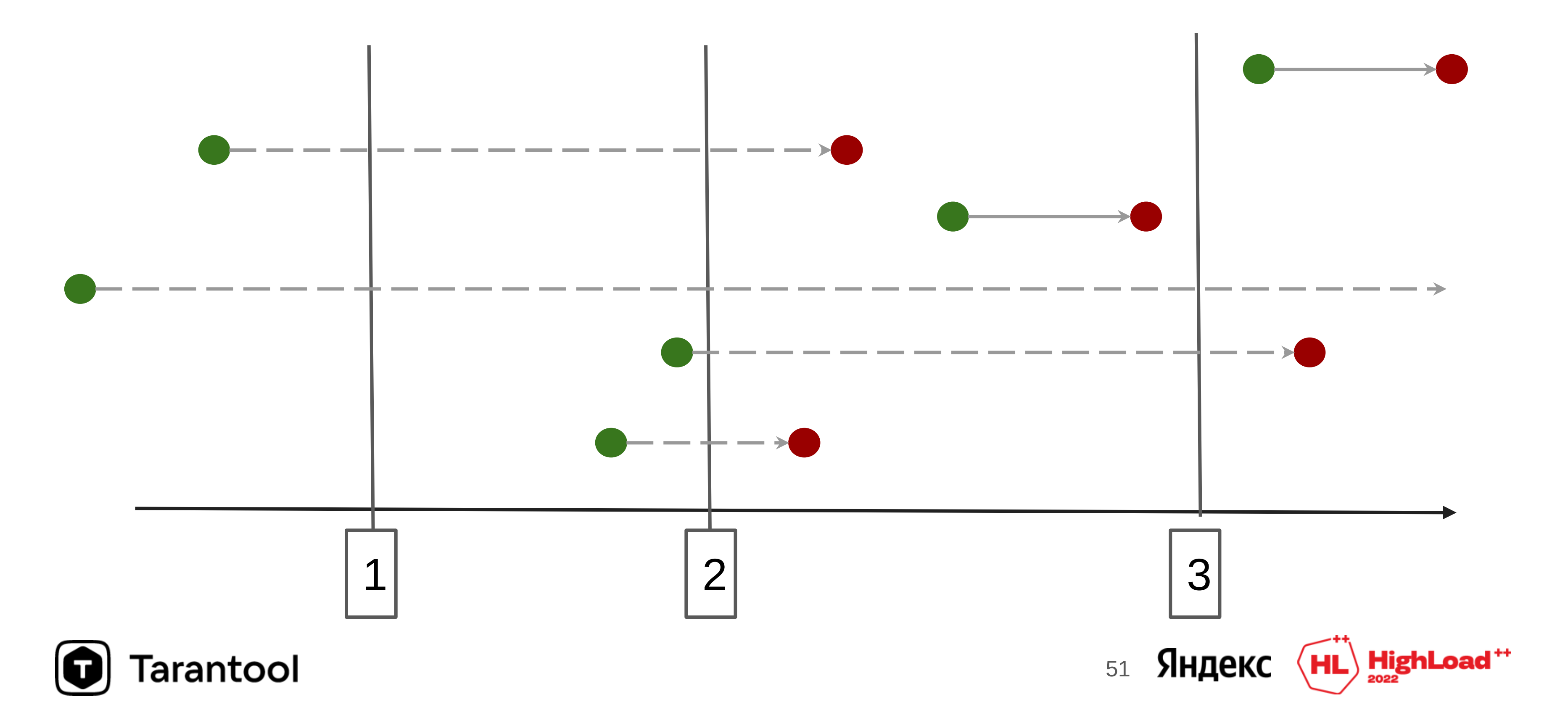
время

читатель версии

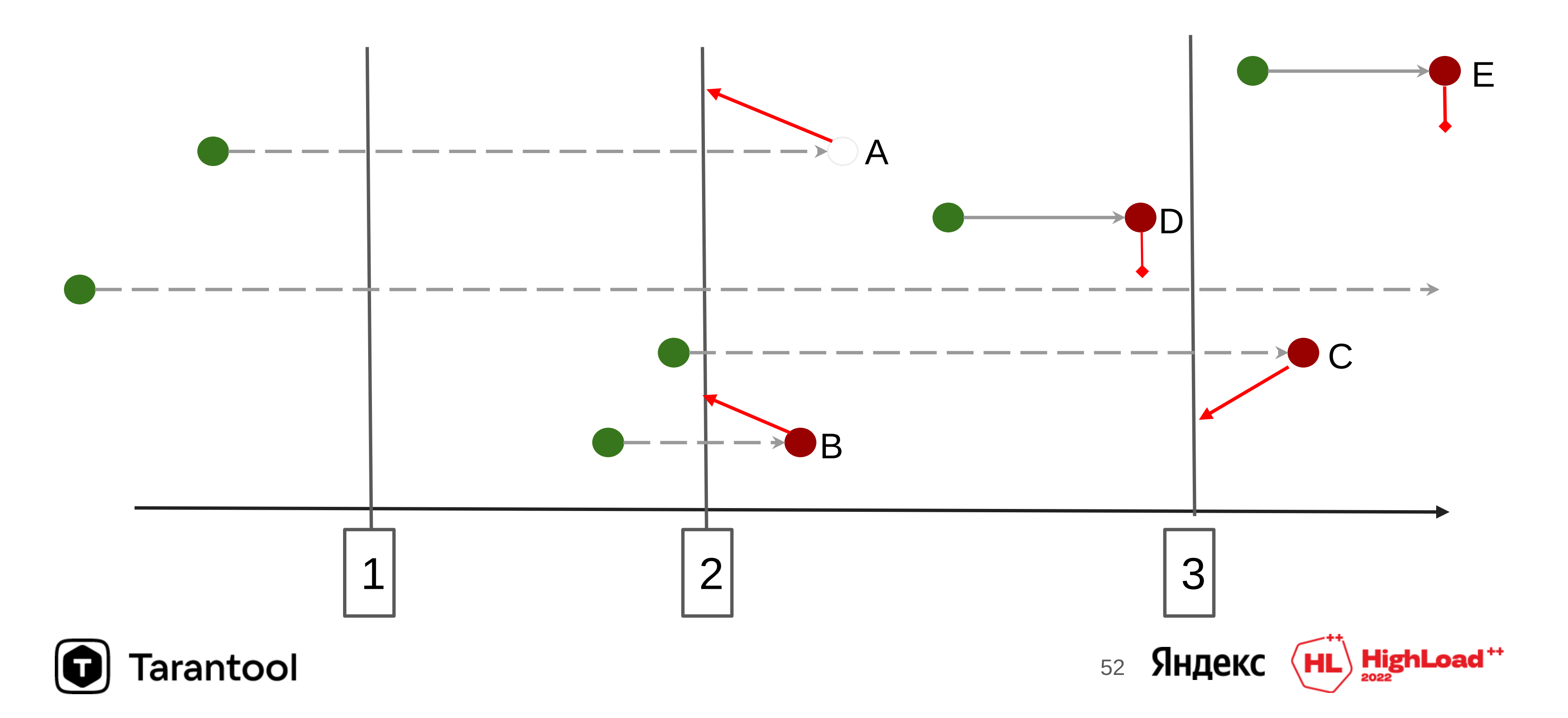




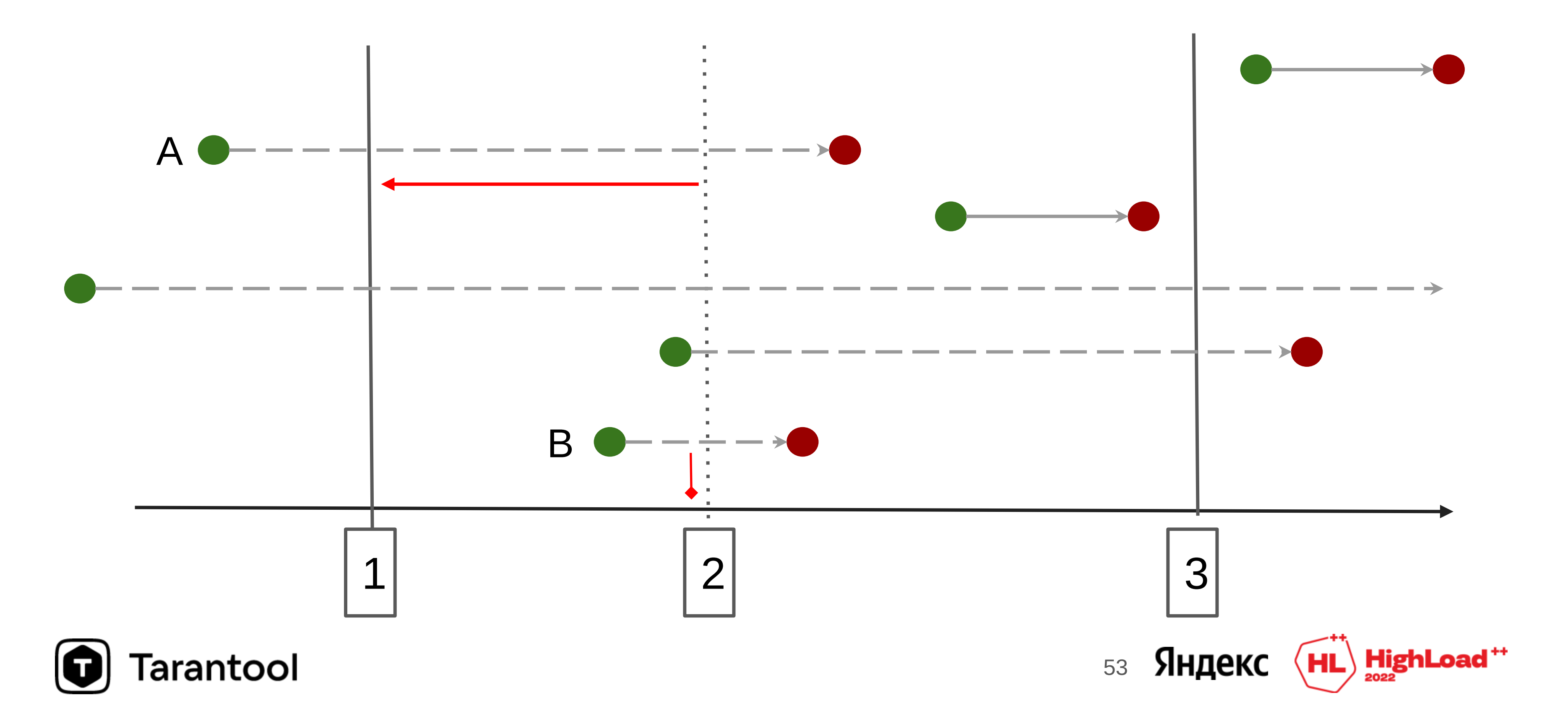
Сохранение копии данных: read view



Сохранение копии данных: read view



Сохранение копии данных: read view



Сохранение копии данных: read view

Основные принципы

- При удалении объект может быть отдан в read view.
- У read view много списков для оптимизации его уничтожения.
- При уничтожении read view какие-то списки удаляем, какие-то переносим в соседний read view.
- Списки интрузивные (перезатирается версия и еще пара не очень нужных полей).
- Сложность удаления read view – O(количество read view).





Read view – пример

```
tarantool> box.space.test:select{}
---
- - [1, 'original']
  - [2, 'another original']
...
tarantool> rv = box.read_view.open()
tarantool> box.space.test:delete{2}
tarantool> box.space.test:insert{3, 'new'}
...
```





Read view – пример

```
tarantool> box.space.test:select{}
---
- - [1, 'original']
  - [3, 'new']
...
tarantool> rv.space.test:select{}
---
- - [1, 'original']
  - [2, 'another original']
...
```





Сохранение копии данных: read view

Итого

- Разные алгоритмы для индексов и данных.
- Отличная алгоритмическая сложность.
- Дешево по памяти.
- Read view доступен из тредов без синхронизации.
- Подходит только для read-only операций.





Сохранение истории изменений

Основные принципы

- В отличие от дисковых БД историю можно делать списком.
- Не реализовывать для каждого типа индекса.
- Не пользуешься – не платишь.
- Алгоритм должен быть алгоритмически простым.





Сохранение истории изменений

Индекс

100, some data

102, more data

101, original data

Сохранение истории изменений

Индекс

100, some data

102, more data

101, new data

101, original data





Сохранение истории изменений

Основные принципы

- В каждом кортеже может стоять бит о том, что есть *история*.
- Связь кортеж → *история* – O(1) всегда.
- История содержит изменения, чтения как конкретно этого кортежа, так и ключей рядом.
- При чтении чистого кортежа - всё просто, грязного - разгребаем историю.
- Считаем, что история невелика.





Сохранение истории изменений

Tarantool: уровень изоляции serializable, но не snapshot

- Пользователю нужен не снапшот на какое-то конкретное время, ему нужна консистентность.
- Чем позже снапшот – тем дешевле и меньше конфликтов.
- Менеджер конфликтов дает плавающий снапшот, фиксирующийся при конфликте.
- Опции транзакции позволяют создавать снапшот из неподтвержденных данных.



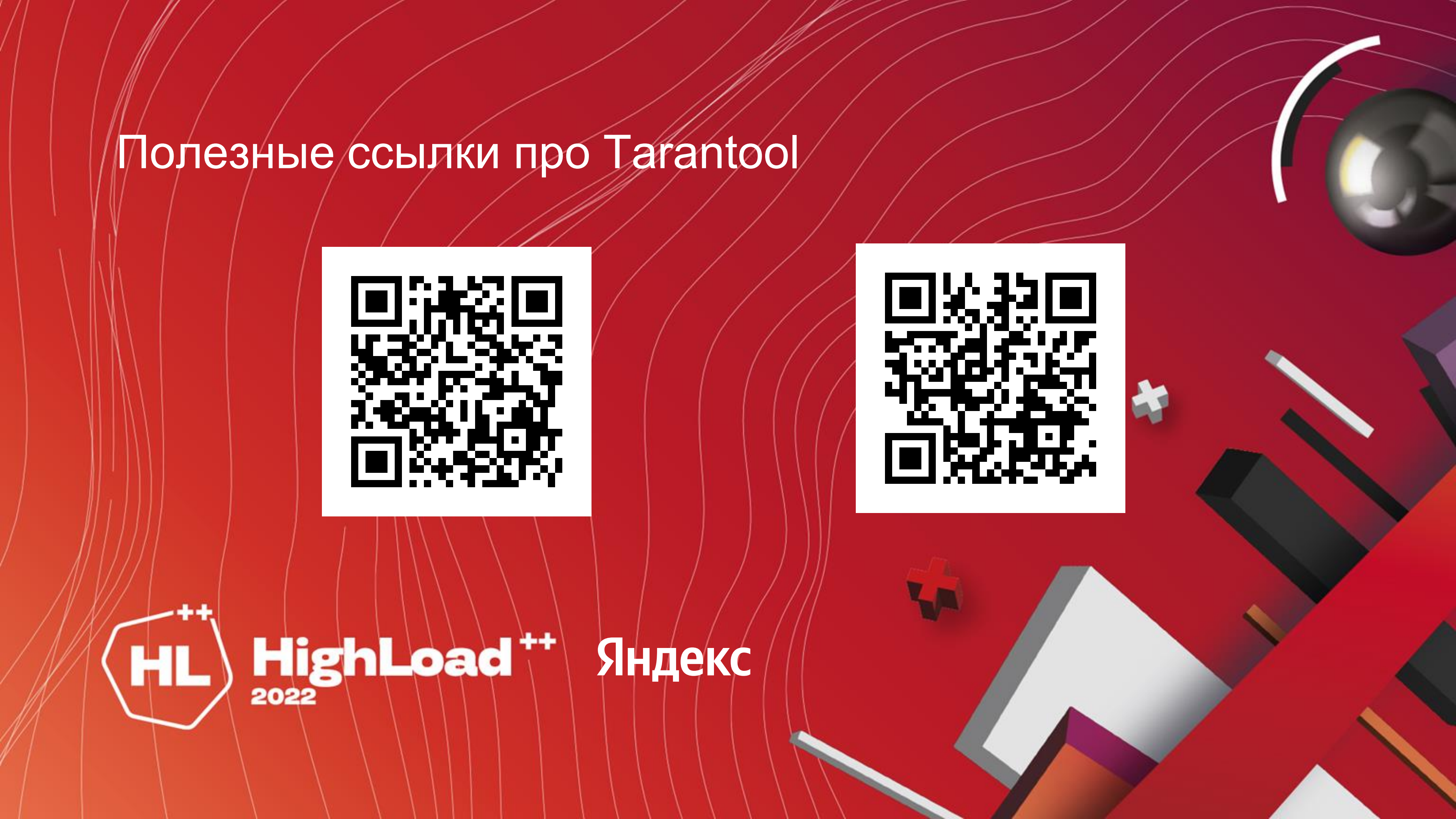


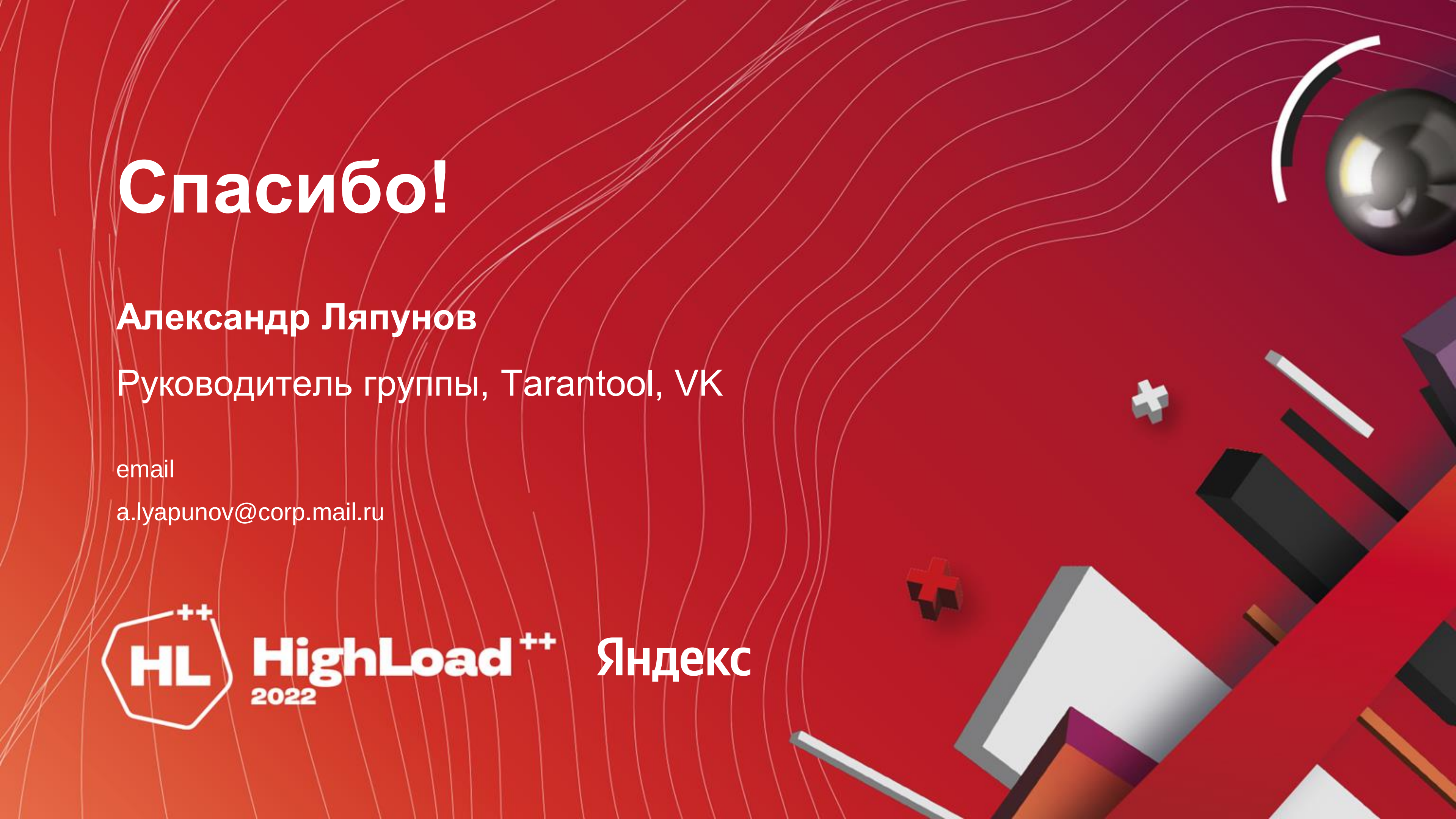
Сохранение истории изменений



Итоги

- MVCC для менеджера транзакций.
- Serializable уровень изоляции.
- Сложность ограничена длиной истории.
- Никакого дополнительного поиска, кроме прыжка кортеж → история (ну и шагов по истории).
- Подробнее про менеджер транзакций - QR-код.


Полезные ссылки про Tarantool
HL
HighLoad++
2022
Яндекс


Спасибо!
Александр Ляпунов
Руководитель группы, Tarantool, VK
email
a.lyapunov@corp.mail.ru
HL
HighLoad++
2022
Яндекс

Обратная связь
и комментарии
по докладу по ссылке



